

Deploying CCXML for Application-Layer Call Control

Eric Burger

CTO of Cantata Technology

Mark Scott

VP Development at Genesys Telecommunication Labs Inc.



cantata[™]
TECHNOLOGY



GENESYS[®]
AN ALCATEL COMPANY

table of contents

Overview	3
Deploying Telephony Applications with VoiceXML	3
Providing Advanced Call Control via CCXML	4
Deploying SIP Applications	6
Bringing it All Together	7
Optimizing the Use of Network Elements	8
Conclusion	9
About the Authors	10

overview

Building and deploying next-generation multimedia services and infrastructure requires the use of the appropriate tools to optimize the productive use of network elements. Call Control eXtensible Markup Language (CCXML) is an application language that provides the advanced application-level call control absent in the widely deployed Voice eXtensible Markup Language (VoiceXML). CCXML is the standard for the setup, monitoring and tear-down of phone calls and conferences, and VoiceXML is the standard for presenting user interfaces that use voice, touch-tones, and speech to interact with the user.

Both CCXML and VoiceXML are valuable standards, and it is important to understand how they are incorporated into network elements. The use of media servers running VoiceXML and SIP application servers running CCXML provide network operators with an efficient model for supporting scalable media services with advanced call control. This whitepaper provides an overview of VoiceXML and CCXML and it describes how they are best deployed to optimize network resources and deliver high-value telephony services.

deploying telephony applications with VoiceXML

VoiceXML is designed for creating interactive dialogs that feature audio and/or video content, synthesized speech, DTMF key press and spoken input. Unlike historical approaches, VoiceXML fully leverages the benefits of speech, allowing for a seamless mix of pre-recorded and synthesized content, as well as for natural language input and mixed initiative. It is based on the Worldwide Web Consortium's (W3C's) Extensible Markup Language (XML), and leverages the Web paradigm for application development and deployment.

Its major goal is to bring the advantages of Web-based development and content delivery to Interactive Voice Response (IVR) applications. Just as HTML is commonly used for creating graphical Web applications, VoiceXML can similarly be used for creating interactive telephony applications.

VoiceXML enables distributed application design by separating each application's user interaction layer from its service logic. This allows the service logic to run on standard Web servers, in many case leveraging existing business logic that is consistent across different access methods. The key difference is that instead of generating HTML content for display in a Web browser, the service logic now generates VoiceXML for delivery via a "voice browser". Most developers find VoiceXML application development at least three times faster than development in traditional IVR environments, in particular because they can leverage their experience with existing languages and environments such as J2EE and .NET. For these reasons, VoiceXML has already been widely adopted within the IVR industry.

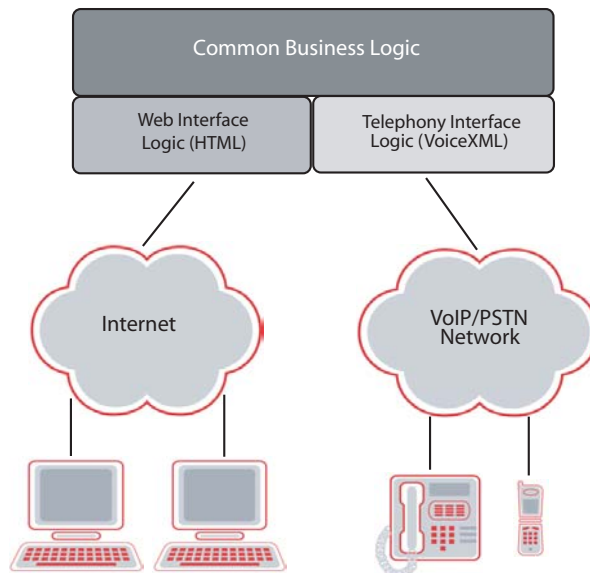


Figure 1: The Web server manages the common business logic, and a browser interprets a presentation language to present the information to a PC or telephone.

providing advanced call control via CCXML

VoiceXML was never designed for advanced call control features. The design goal for VoiceXML was as a dialog language; it manages the presentation of media and the handling of input in the context of a dialog. It uses a model based on the concept of a form, and defines rules for how these forms control interaction with the user. This model works well for telephony user interfaces; however, it is ill-suited to deal with call control concepts and events that can occur at any time and that cannot be easily mapped to the form construct that is central to VoiceXML.

The fundamental and proper focus on the user interface taken by VoiceXML makes it very difficult to deliver call control within the confines of the language itself. Thus, the W3C created the CCXML initiative to be an orthogonal language to VoiceXML, specifically intended to meet the needs of call control applications. By necessity, many vendors have previously developed proprietary call control extensions for VoiceXML; CCXML provides a standards-based alternative. CCXML is currently under active development by the call control subgroup of the W3C Voice Browser Working Group and is nearing completion.

CCXML is a declarative, XML-based markup language for writing call control logic and applications, and it is structured around the handling of events that occur within the context of CCXML sessions. CCXML defines a high-level, abstract model that can be implemented across many different types of underlying networks and protocols. CCXML applications implement behavior through manipulating connections, conferences and dialogs. Dialogs are the means by which a CCXML application makes use of VoiceXML functionality.

CCXML sessions are the context in which CCXML documents are executed. CCXML documents define behavior of a session by controlling how a server responds to events within the context of that session. A CCXML document consists of initialization content (scripts) that are run when a

document is first loaded, and a set of transitions that define how an application responds to events.

Sessions are created in response to incoming calls received by a CCXML platform, and they can also be created through other means such as HTTP in order to place outbound calls. As part of being created, a CCXML session fetches an initial CCXML document that defines how the session will react to events. Sessions implement services by:

- Manipulating connections, conferences and dialogs
- Interfacing with other sessions
- Handling external events

Sessions own connections and dialogs, and can interact with them through language-defined elements and events. Sessions similarly interact with conferences that may be local or global, and events can be sent to and received from other sessions and external sources.

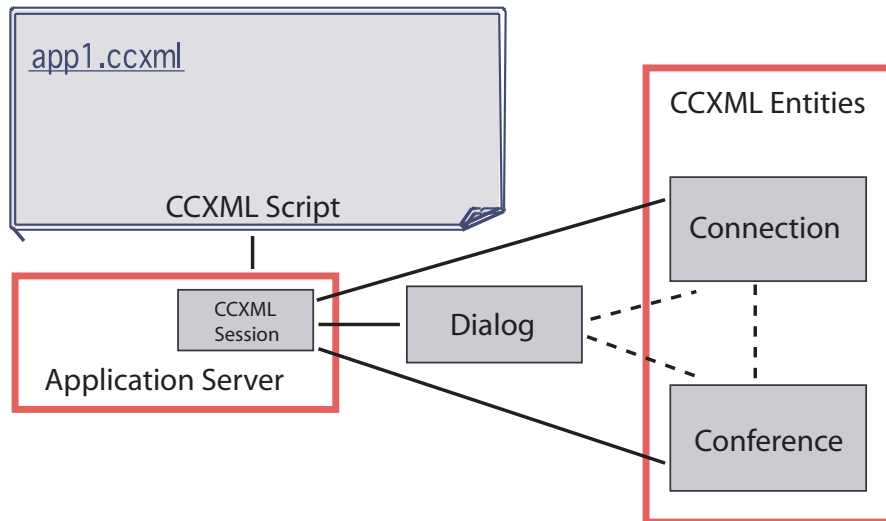


Figure 2: CCXML runs on an application server and allows applications to manipulate connections, conferences and dialogs.

CCXML connections represent calls being handled by the CCXML platform. Connections represent a Session Initiation Protocol (SIP) session established to the platform, which might originate from a media gateway, softswitch or some other SIP element routing a call to the CCXML platform to access a particular service. CCXML provides a way to:

- Answer, reject or redirect incoming connections
- Initiate outbound connections
- Disconnect or transfer established connections

CCXML dialogs model an interactive dialog, most frequently embodied as a VoiceXML session

that is under the control of a CCXML session. CCXML dialogs can be used to collect information from a caller, play hold music, provide messaging access or perform any number of other functions. Applications are able to start or terminate dialogs, and they can receive information collected by a dialog. Dialogs are typically written using VoiceXML, but other dialog environments are possible, such as simple announcements using RFC 4240, the Network Announcement Protocol (NETANN).

Connections and dialogs can be joined to one another, allowing callers to speak to each other and allowing callers to be connected to a voice dialog. Joining two connections or dialogs creates a bridge, which represents the flow of media from one connection or dialog to another. Bridges can be either full duplex (two-way) or half duplex (one-way). A connection or dialog can only listen to a single source, but can transmit to any number of listeners.

CCXML conferences allow multiple connections and/or dialogs to be joined together so that multiple sources can be heard by a single listener. Conferences have any number of inputs and have a single logical output. Conferences represent mixing capabilities of the media platform and can be created and destroyed by an application. Conferences can be used locally to a session or globally across multiple sessions.

CCXML is just one of a number of available languages for writing call control logic; whether it is the most appropriate for a given application depends on the needs of that specific application. The following are some of the key aspects of CCXML that can assist in determining whether it is the most appropriate approach:

- A high-level, XML-based programming model is very easy to learn and allows applications to be created very quickly.
- A Web-based development model allows the leveraging of experience with existing languages and environments such as J2EE and .NET, and the re-use of software components.
- A Media-centric functionality is modeled by the language; interactions with conference bridges and VoiceXML dialogs are independent of the protocols used to achieve this.

deploying SIP applications

CCXML functions as a SIP application server, and both CCXML and VoiceXML interact with an HTTP application server, which is where the real service/business logic resides. VoiceXML provides a means for an application to specify how a system interacts with a user using Real-Time Protocol (RTP), and CCXML provides a means for an application to specify how to interact with the user from a SIP perspective.

For certain applications such as conferencing, additional functionality not contained in SIP is required. For example, SIP provides the basic tools for simple conferencing functionality but more advanced call control functionality is required for large-scale applications. Two complementary protocols can be used in conjunction with SIP for IP conferencing services: NETANN, which provides simple announcement and basic conferencing services, and Media Server Control Mark-up Language (MSCML).

MSCML is used in conjunction with SIP to enable the delivery of advanced multimedia conferencing services over IP networks. It provides the conferencing control functions that

allow application servers to make use of enhanced features available on media servers. MSCML adds value to SIP applications by providing developers with the tools to offer more advanced features for SIP conferencing applications. MSCML enables new converged applications, mixing the rich media IVR experience of VoiceXML with enhanced conferencing facilities.

NETANN is a general SIP convention for initiating specific services on a multifunction media server. Likewise, MSCML is a control protocol specific to enhanced conferencing. Application servers use NETANN to access basic application functions, such as announcements, basic media mixing and the prompting and collecting of user information from the media server, while MSCML provides developers with a framework to add enhanced conferencing functionality. For example, MSCML uses the SIP conventions described by NETANN to determine when to:

- Start an enhanced conference
- Add a leg to a conference
- Drop a leg from a conference
- Create complex conference topologies
- Terminate a conference

bringing it all together

NETANN, MSCML, VoiceXML and CCXML are complementary technologies:

- NETANN provides simple announcement and basic conferencing services as well as advanced session initiation capability
- MSCML runs on media servers and provides enhanced conferencing services
- VoiceXML enables enhanced IVR
- CCXML provides advanced application-level call control

Developers build robust call processing applications incorporating advanced conferencing and rich user interaction using VoiceXML and CCXML, with the application logic running on standard web (HTTP) application servers. The SIP application server runs CCXML, and uses a combination of SIP, NETANN, MSCML and VoiceXML to access functionality provided by the media server used in executing call control logic written in CCXML.

When CCXML applications invoke interactive dialogs, NETANN is used to run VoiceXML applications on the media server. If CCXML applications access conferencing functionality, then NETANN and MSCML may be used to access conferencing capabilities provided by the media server.

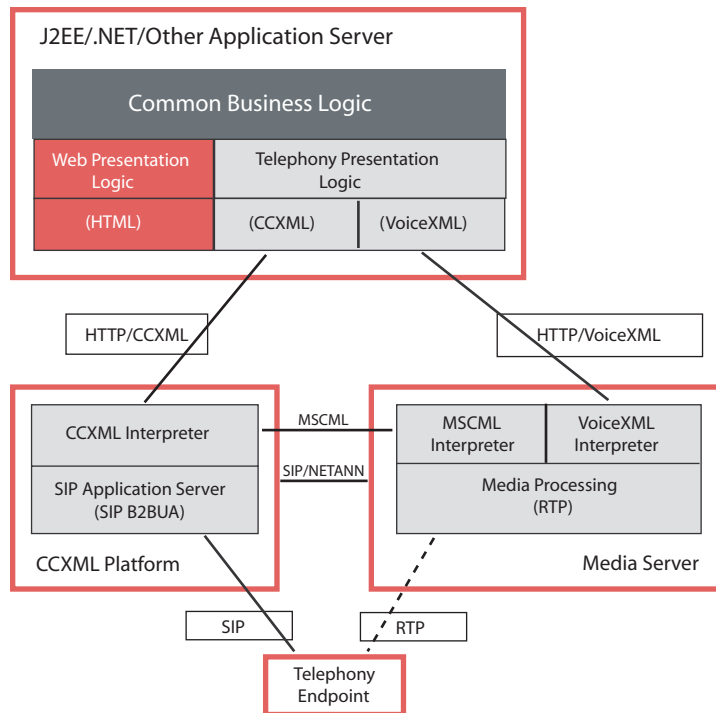


Figure 3: CCXML and VoiceXML interact closely to enable enhanced conferencing and IVR functionality

optimizing the use of network elements

There is a purity of this deployment model that dramatically impacts the cost and efficiencies of deploying IVR and conferencing services. Each network element is designed for a single, specific purpose and they interact over standardized protocols to support the rapid development and smooth delivery of enhanced services. This separation enables independent enhancement and development of network elements.

By deploying CCXML on SIP application servers and VoiceXML on media servers, each of these platforms can be used to efficiently support next-generation services. Deploying both on a single server would be an inefficient use of resources. For example, CCXML applications typically require the use of VoiceXML for dialog services for only a limited part of the overall call; the ratio of CCXML to VoiceXML resources is highly application dependent, and may change over time.

The separation of functionality between SIP application servers and media servers allows organizations to optimize each to accommodate business requirements. A single media server would likely support several separate, distinct and independent SIP application servers, and network operators could more efficiently and cost effectively scale the network to accommodate new users or applications. This separation becomes increasingly important as services scale, since the relationship between media servers and application servers becomes many-to-many.

For example, consider the use of a prepaid calling application using CCXML. The application server running CCXML handles an incoming call and immediately routes it over to the media server running VoiceXML to identify the user and inform him-or-her of the account balance. The application server-driven by CCXML then places the outbound prepaid call and connects users. At this point VoiceXML and the media server are no longer involved in the call. The CCXML session is still involved to monitor credit card utilization. Upon credit expiry, the CCXML script can bring the media server back into the call for the user to recharge the account via a voice interaction specified by VoiceXML.

If both CCXML and VoiceXML were combined on a single server, the use of resources would be tremendously inefficient, since VoiceXML would be rarely used once the call was established. Applications using IVR similarly require the phased usage of media-and it would therefore be similarly inefficient to integrate both CCXML and VoiceXML onto a single server to support them.

conclusion

CCXML was created to accelerate the development time of applications for the telephone, such as conferencing, call center integration and intelligent call routing capabilities such as find-me/follow-me services. It was developed as an adjunct to VoiceXML. CCXML runs as a SIP application server, and both CCXML and VoiceXML interact with an HTTP application server to access the service and business logic of an application. CCXML provides a means for an application to specify SIP-based management of calls, and VoiceXML provides a means for an application to specify RTP-based interactions with the end-user.

The standardization of call control within an XML framework helps developers leverage their Web programming backgrounds and reduce the costs and time of developing new services, and it allows organizations to more swiftly and cost effectively scale new services as they evolve over time.

about the authors

Mark Scott, VP Development of Genesys

Mark Scott is VP Development at Genesys Telecommunication Labs Inc., which he joined through its acquisition of VoiceGenie, where Mark served as CTO since 2002. At Genesys, Mark continues to focus on the capabilities, architecture, and design of standards-based voice platform products. Mark holds an M.A.Sc in Computer Engineering from the University of Waterloo, and is also the inventor of 5 patents in the area of VoIP

Eric Burger, CTO of Cantata Technology

Eric Burger is CTO of Cantata Technology. Eric joined Cantata through the combination of SnowShore Networks (of which he was a founder), Brooktrout Technology and Excel Switching. The SnowShore IP Media Server™ is the industry's premier software-based, carrier-grade IP media server. It leverages the simplicity, openness and flexibility of SIP, VoiceXML and MSCML to provide a cost-effective and scalable IP media server solution. Eric is responsible for setting the technology direction for Cantata. He serves in several industry standards forums as a participant and in leadership positions, including the IETF, IEEE and W3C. In addition, he is a member of the Board of Directors of the SIP Forum and IMS Forum. Eric has over 15 patents issued and pending, and holds degrees from MIT, IIT and the Catholic University of Leuven.



cantata[™]
TECHNOLOGY

Corporate Headquarters

410 First Avenue
Needham, MA 02494
USA

Tel: +1 (781) 449-4100

Fax: +1 (781) 449-9009

Email: info@cantata.com

Cantata Technology maintains multiple locations worldwide in North America, Asia and Europe.

www.cantata.com



GENESYS[®]
AN ALCATEL COMPANY

Corporate Headquarters

2001 Junipero Serra Blvd,
Daly City, CA 94014
USA

Tel: 1-888-GENESYS

Fax: 1-650-466-1260

www.genesyslab.com